Model-Specific Registers

Transmeta Corporation
3940 Freedom Circle
Santa Clara, CA 95054

Title: Model-Specific Registers
Date: 2002/04/10
Purpose: This document describes the Model-Specific Registers (MSRs) that are supported by Transmeta processors, as well as the instructions that can be used to access those Model-Specific Registers.

Model-Specific Registers

Property of:
Transmeta Corporation
3940 Freedom Circle
Santa Clara, CA 95054
USA
(408) 919-3000
http://www.transmeta.com

Introduction

The Model-Specific Registers (MSRs) are designed to allow the programmer to observe and control the behavior of processor-specific details. Three instructions are provided to access the MSRs, and the CPUID instruction can be used to determine whether MSRs are supported or not. Transmeta processors do support MSRs, including the Time Stamp Counter (TSC).

Note that some MSRs are only implemented if the corresponding CPUID feature flag is set to 1. For example, the TSC MSR is supported if the TSC feature flag is set to 1.

In general, MSRs are 64-bit wide; though not every MSR implements all 64 bits. Reserved bits read as 0, while writing a 1 into a reserved bit causes a #GP(0) exception. Reserved bits are indicated by gray shading in the MSR layout figures.

The RDMSR and WRMSR instructions can be used to read and write MSR values, while the RDTSC instruction can be used to read the TSC value. Trying to read from or write to a reserved MSR causes a #GP(0) exception.

The RDMSR and WRMSR instructions are privileged, and in addition WRMSR is serializing. They require the MSR index as an input value, in register ECX. The RDMSR instruction will return the current MSR value in EDX:EAX, while WRMSR requires the desired MSR value as an input in EDX:EAX. It should be noted that the MSRs differ from processor to processor, and should therefore be used with caution.

The RDTSC instruction is only privileged if CR4.TSD is set to 1. It expects no input values, and returns the current TSC value in EDX:EAX. Programmers should refrain from using RDMSR to read the TSC, as the actual MSR that holds the TSC might be implemented at a different index in the future.

Intel-compatible MSRs

Index 0000_0010h – Time Stamp Counter (TSC)

63	TSC (high)	32
31	TSC (low)	0

Support for this MSR is indicated by a standard feature flag (TSC) returned by the CPUID instruction.

The 64-bit TSC value increments at the rate of the nominal processor core clock frequency, ie. that rate remains constant even if the actual processor core clock frequency changes (eg. due to LongRun).

Note that only bits 31…0 can be written; bits 63..32 are zero-extended on writes.

In addition, programmers must keep the non-deterministic instruction timing of Transmeta processors in mind, when the TSC is used for instruction timing purposes.

Index 0000_0119h – Processor Serial Number disable (PSN_DISABLE)

63	reserved					32
31	reserved	22	21	20	reserved	0

Support for this MSR is indicated by a standard feature flag (PSN) returned by the CPUID instruction.

Setting bit 21 to 1 disables the PSN. The bit can not be cleared by WRMSR once it has been set to 1; only a processor RESET clears the bit and re-enables the PSN.

Note that OEMs have the option to permanently disable the PSN. In that case the bit will always be set.

Index 0000_0174h – SYSENTER/SYSEXIT base selector (SEP_SEL)

63	ignored	32
31	scratch bits	16	15	base selector	0

Support for this MSR is indicated by a standard feature flag (SEP) returned by the CPUID instruction.

This is the 16-bit selector that is used by the SYSENTER and SYSEXIT instructions. The upper 32 bits are ignored during WRMSR, and always read as zero. Note that bits 31…16 are scratch bits, ie. they hold their value but have no function.

Index 0000_0175h – SYSENTER ESP (SEP_ESP)

63	ignored	32
31	ESP	0

Support for this MSR is indicated by a standard feature flag (SEP) returned by the CPUID instruction.

This is the 32-bit ESP value that is used by the SYSENTER instruction. The upper 32 bits are ignored during WRMSR, and always read as zero.

Index 0000_0176h – SYSENTER EIP (SEP_EIP)

63	ignored	32
31	EIP	0

Support for this MSR is indicated by a standard feature flag (SEP) returned by the CPUID instruction.

This is the 32-bit EIP value that is used by the SYSENTER instruction. The upper 32 bits are ignored during WRMSR, and always read as zero.

Transmeta-specific MSRs

The 8086_xxxxh range of MSRs is divided into two portions. The intention is to establish one common standard for MSRs employed by all x86 vendors. This is driven by the fact that some vendors have been using a variety of different MSRs to achieve the same results, thus making BIOS and OS support, as well as the creation of utilities harder than they need to be.

- **8086_0000…8086_7FFFh** **common to all processors, documented in public**
  - 8086_0000…8086_000Fh CPUID-related MSRs (all Transmeta processors)
- **8086_8000…8086_FFFFh** **vendor-specific, may or may not be documented in public**
  - 8086_8010…8086_8017h LongRun MSRs
  - 8086_8018…8086_801Fh LongRun Table Interface MSRs, part 1/2 (processor settings)
  - 8086_8030…8086_803Fh LongRun Table Interface MSRs, part 2/2 (memory settings)

Index 8086_0000h – CPUID type. family, model and stepping (CPUID_TFMS)

63	reserved	32
31	reserved or extended (see note)	16	15	T	12	11	F	8	7	M	4	3	S	0

This MSR allows to modify the type (bits 15…12), family (bits 11…8), model (bits 7…4), and stepping (bits 3…0) values, which are reported by the CPUID instruction, function 0000_0001h, register EAX. Up to and including CMS 4.2, bits 31…16 are reserved. Starting with CMS 4.3 they allow to modify the upper 16 bits of the value reported by CPUID, function 0000_0001h, register EAX.

Index 8086_0001h – CPUID vendor ID string, part 1/3 (CPUID_VND1)
Index 8086_0002h – CPUID vendor ID string, part 2/3 (CPUID_VND2)
Index 8086_0003h – CPUID vendor ID string, part 3/3 (CPUID_VND3)

63	reserved	32
31	VND1	0

63	reserved	32
31	VND2	0

63	reserved	32
31	VND3	0

These three MSRs allow to modify the vendor ID string that is reported by the CPUID instruction, function 0000_0000h, registers EBX, EDX, and ECX. The MSRs form a contiguous string, and map to the CPUID return values as follows: CPUID_VND1=EBX, CPUID_VND2=EDX, and CPUID_VND3=ECX.

Index 8086_0004h – CPUID feature flags mask (CPUID_MASK)

63	reserved	32
31	MASK	0

This MSR allows to mask out the feature flags that are reported by the CPUID instruction, function 0000_0001h, register EDX. If a bit is set to 0 in this mask, then the corresponding feature flag will be forced to 0; otherwise the corresponding feature flag will be reported without changes.

Note that CMS 4.1 forces bit 8 to 0 by default, due to an issue with the Microsoft Windows NT operating system.

Index 8086_8010h – LongRun control and status register (LONGRUN)

63	reserved	39	38	upper bound	32
31	reserved	7	6	lower bound	0

Support for this MSR is indicated by a Transmeta feature flag (LongRun) returned by the CPUID instruction.

When LongRun is available, this MSR allows to select and query the current performance window.

Bits 63…32 hold a number which must be no larger than 0000_0064h (100d, or 100%); this number represents the upper boundary of the performance window.

Bits 31…0 hold a number which must be no larger than 0000_0064h (100d, or 100%); this number represents the lower boundary of the performance window.

Use WRMSR to specify the desired performance window. Specifying boundary values outside the 0…64h range causes a #GP(0) exception, as does specifying a lower boundary that is greater than the upper boundary.

A performance window can be achieved by specifying an upper and lower boundary. A fixed performance level can be achieved by specifying the same value for both the upper and lower boundary.

Setting both boundaries to 100% effectively disables LongRun, and results in maximum performance, but shorter battery lifetime. By contrast, setting both boundaries to 0% results in minimum performance, but best battery lifetime.

The processor will round the specified values up or down to the nearest value, depending on how many performance levels are supported. If both directions of rounding are possible, then the processor will round up.

Use RDMSR to query the current performance window. Notice that the processor returns rounded values; thus they may or may not match the most recently written values.

Additional LongRun status information can be queried via CPUID, function 8086_0007h. For details, please refer to the Processor Recognition documentation.

Index 8086_8011h – LongRun flags register (LONGRUN_FLAGS)

63	reserved	32
31	reserved	1	0

Support for this MSR is indicated by a Transmeta feature flag (LongRun) returned by the CPUID instruction.

Bit 0 controls whether LongRun is in “economy mode” (0) or in “performance mode” (1). In “economy mode” the processor attempts to maximize battery lifetime, by powering down aggressively. By contrast, in “performance” mode it attempts to maximize performance, by powering down conservatively.

Index 8086_8018h – LongRun Table Interface readout (LRTI_READOUT)

63	maximum supported LongRun level	32
31	currently selected LongRun level	0

Support for this MSR is indicated by a Transmeta feature flag (LRTI) returned by the CPUID instruction.

This MSR allows to query how many LongRun levels are supported, and it allows to select and query which level can currently be read out. The upper 32 bits are ignored on WRMSR; trying to select a non-existing level causes #GP(0).

The levels are enumerated starting with zero; level zero is always the level with the highest performance.

Index 8086_8019h – LongRun Table Interface performance index (LRTI_PERF)
Index 8086_801Ah – LongRun Table Interface voltage and frequency (LRTI_VOLT_MHZ)
Index 8086_801Bh – LongRun Table Interface memory divisors (LRTI_DIV_MEM)
Index 8086_801Ch – LongRun Table Interface I/O divisors (LRTI_DIV_IO)
Index 8086_801Dh – LongRun Table Interface gate delay (LRTI_GATE)

63	reserved	32
31	performance index (%) for selected level	0

63	voltage (mV) for selected level	32
31	frequency (MHz) for selected level	0

63	DDR divisor for selected level	32
31	SDR divisor for selected level	0

63	reserved	32
31	PCI divisor for selected level	0

63	reserved	32
31	measured gate delay (fs) for selected level	0

Support for these MSRs is indicated by a Transmeta feature flag (LRTI) returned by the CPUID instruction.

These MSRs return the values that correspond to the selected LongRun level (see LRTI_READOUT MSR). They are read-only; trying to write to any of them causes a #GP(0) exception. If a field is not applicable, then a value of zero is reported.

Index 8086_8030h – LongRun Table Interface SDR MISC (LRTI_SDR_MISC)
Index 8086_8031h – LongRun Table Interface SDR TIF (LRTI_SDR_TIF)
Index 8086_8032h – LongRun Table Interface SDR MRS (LRTI_SDR_MRS)
Index 8086_8033h – LongRun Table Interface SDR LBT (LRTI_SDR_LBT)
Index 8086_8034h – LongRun Table Interface DDR MISC/TIF (LRTI_DDR_MISC_TIF)
Index 8086_8035h – LongRun Table Interface DDR MRS0 (LRTI_DDR_MRS0)
Index 8086_8036h – LongRun Table Interface DDR MRS1_MRS2 (LRTI_DDR_MRS1_MRS2)
Index 8086_8037h – LongRun Table Interface PFTIMING (LRTI_PFTIMING)

63	reserved	32
31	SDR_MISC for selected level	0

63	SDR_TIF1 for selected level	32
31	SDR_TIF0 for selected level	0

63	SDR_MRS1 for selected level	32
31	SDR_MRS0 for selected level	0

63	SDR_LBT1 for selected level	32
31	SDR_LBT0 for selected level	0

63	DDR_MISC for selected level	32
31	DDR_TIF for selected level	0

63	reserved	32

31	DDR_MRS0 for selected level	0

63	DDR_MRS1 for selected level	32
31	DDR_MRS2 for selected level	0

63	reserved	32
31	PFTIMING for selected level	0

Support for these MSRs is indicated by a Transmeta feature flag (LRTI) returned by the CPUID instruction.

These MSRs return the values that correspond to the selected LongRun level (see LRTI_READOUT MSR). They are read-only; trying to write to any of them causes a #GP(0) exception.

END